INTERNATIONAL STANDARD

ISO/IEC 23007-1

First edition
2010-11-01

AMENDMENT 1
2012-01-15

Information technology — Rich media user interfaces —

Part 1: **Widgets**

AMENDMENT 1: Widget extensions

Technologies de l'information — Interfaces d'utilisateur au support riche —

Partie 1: Widgets

AMENDEMENT 1: Extensions de widget

Reference number
ISO/IEC 23007-1:2010/Amd.1:2012(E)

ISO copyright office
Case postale 56 • CH-1211 Geneva 20
Tel. + 41 22 749 01 11
Fax + 41 22 749 09 47
E-mail copyright@iso.org
Web www.iso.org

Published in Switzerland

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2.

The main task of the joint technical committee is to prepare International Standards. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

Amendment 1 to ISO/IEC 23007-1:2010 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 29, *Coding of audio, picture, multimedia and hypermedia information*.

This Amendment introduces extensions for advanced widgets and improves interoperability between widget managers.

Information technology — Rich media user interfaces —

Part 1:
Widgets

AMENDMENT 1: Widget extensions

At the end of 6.1, add:

In order to be discoverable by other widget managers and recognizable as an MPEG-U conformant implementation, an MPEG-U widget manager shall advertise a service of type "`urn:mpeg:mpeg-u:standard-service:widget-manager`" on all the service and discovery protocols that it supports to manage the services offered or required by the widgets.

Using this service, widget managers shall support the reception of a "`startWidget`" message, with either a "`widgetUrl`" or a "`widgetUUID`" parameter, as strings, and optionally with a "`widgetContext`", a string containing the widget context information as defined in Clause 11. The widget manager shall reply to this message with an integer parameter called "`errorCode`", whose value is 0 if the widget was successfully loaded and any other value if the loading failed.

Using this service, widget managers shall support the reception of a "`requestCapabilitiesList`" message with no parameter. The widget manager shall reply with a message containing a description of the processing capabilities of the widget manager. The description shall be formatted as a string in the "`capabilities`" parameter and may be empty. The format of this parameter may be given in the "`capabilitiesType`" in the form of a mime type. If this parameter is omitted, the default value is "application/xml".

Using this service, a widget manager can also query and request migration of widgets from another widget manager. The list of widgets can be retrieved using the `listWidgets` message, which allows a remote widget manager to query the list of possible widgets from the target widget manager. The reply contains a list of (numerical) codes, a list of widget names as a space-separated list of quoted strings, and an optional list of space-separated list of UUIDs.

NOTE The selection of the widgets to be listed is implementation specific: a widget manager may decide to only list some activated widgets or all widgets installed, whether activated or not.

A widget manager can then request the migration of a widget from the target widget manager using the `getWidget` message. This message instructs the target widget manager to trigger the migration of the widget whose code is provided. The target widget manager then sends a `startWidget` message to the widget manager originating this message. The following error codes are defined:

- errorCode 0: success
- errorCode 1: requested widget is no longer available
- errorCode 2: target widget manager denied sending of the widget
- errorCode 3: any other error.

NOTE The target widget manager may refuse to migrate a widget for security reasons. The mechanisms used by such a widget manager to certify other widget managers are outside the scope of this specification.

NOTE The default value for capabilitiesType is 'application/xml' as most capability description standards use XML documents (e.g W3C CC/PP, MPEG-21 UED).

The following code uses the syntax described in Clause 10 to describe this service. But, a widget shall not have this interface in its manifest and widget managers shall fail to load any widget, which has such interface in its manifest.

```
<mw:interface type="urn:mpeg:mpeg-u:standard-service:widget-manager"
              serviceProvider="true" multipleBindings="true">
      <mw:messageIn name="startWidget">
          <mw:input name="widgetUrl" scriptParamType="string"/>
          <mw:input name="widgetUUID" scriptParamType="string"/>
          <mw:input name="widgetContext" scriptParamType="string"/>
          <mw:output name="errorCode" scriptParamType="number"/>
      </mw:messageIn>
      <mw:messageIn name="requestCapabilitiesList">
          <mw:output name="capabilitiesType" scriptParamType="string"/>
          <mw:output name="capabilities" scriptParamType="string"/>
      </mw:messageIn>
      <mw:messageIn name="listWidgets">
          <mw:output name="widgetCodes" scriptParamType="string"/>
          <mw:output name="widgetNames" scriptParamType="string"/>
          <mw:output name="widgetIdentifiers" scriptParamType="string"/>
      </mw:messageIn>
      <mw:messageIn name="getWidget">
          <mw:input name="widgetCode" scriptParamType="number"/>
          <mw:output name="errorCode" scriptParamType="number"/>
      </mw:messageIn>
</mw:interface>
```

At end of 7.2 Widget life cycle, add:

The widget manager shall maintain a set of known widgets: a known widget is a widget that the widget manager has checked that it can be rendered. One use of this set is to search for a component matching given interfaces, as described in 10.20.

NOTE The widget manager may provide more services on known widgets, such as displaying a catalog of icons of known widgets.

After the scriptParamType attribute description in 10.18, add:

Possible values are boolean, number, string, stringarray. The type stringarray is reserved for use within predefined communications with the widget manager.

At the end of 10.19, add:

`mw:scriptParamType`

When used as a parameter within a mapping to a script function call, this attribute defines the type of the parameter on which this input is mapped. Possible values are boolean, number, string, stringarray. The type stringarray is reserved for use within predefined communications with the widget manager.

In 10.20 The <mw:component> element, replace the sentence:

Upon loading of this manifest, the widget manager shall get the manifest of the component widgets referenced by the mw:src attribute or try to find a widget that matches the interfaces declared by the <mw:requiredInterface> elements, without actually activating the component widget.

with:

Upon loading of this manifest, the Widget Manager shall get the manifest of the component widgets referenced by the mw:src attribute or try to find in its set of known widgets as defined in 7.2, a widget that matches the interfaces declared by the <mw:requiredInterface> elements, without actually activating the component widget.

At the end of 10.20, add:

`mw:activationTarget`

Optional. This attribute describes when and how the widget manager activating the component, called the local widget manager, may forward it, with the following possible values:

- localOnly (default): the local widget manager shall not forward this component to other widget managers.
- localAndAllRemote: the local widget manager shall activate this component and forward it to other remote widget managers.
- localOrSingleRemote: the local widget manager shall activate this component or, if local activation failed, shall forward it to a single remote widget manager.
- allRemote: the local widget manager shall not activate this component and shall forward it to all the remote widget managers.
- singleRemote: the local widget manager shall not activate this component and shall forward it to a single remote widget manager.

In the cases localOrSingleRemote and singleRemote, the widget manager shall try to load the component on all possible remote widget managers one by one and shall stop as soon as one load is successful. In the cases localAndAllRemote and allRemote, the widget manager shall try to load the component on all possible remote widget managers.

At the end of 10.22, add:

```
<messageOut name="registerWidgetByURL">
    <output name="url" scriptParamType="string"/>
    <input name="returnCode" scriptParamType="number"/>
</messageOut>
```

This message is sent by a Widget to request that the Widget Manager adds the widget found at the given “url” parameter to its set of known widgets. The Widget Manager shall reply using a “returnCode” parameter whose value is: 1 for failure of the request and 0 for success.

```
<messageOut name="activateWidgetByURL">
    <output name="url" scriptParamType="string"/>
    <input name="returnCode" scriptParamType="number"/>
</messageOut>
```

This message is sent by a Widget to request that the Widget Manager activates the widget found at the given “url” parameter. The activated widget is not added to the set of known widgets. The Widget Manager shall reply using a “returnCode” parameter whose value is: 1 for failure of the request and 0 for success.

```
<messageOut name="migrateComponent">
    <output name="componentId" scriptParamType="string"/>
    <output name="targetCode" scriptParamType="string"/>
    <input name="returnCode" scriptParamType="number"/>
</messageOut>
```

This message is sent by a Widget to request that the Widget Manager migrates the component widget with the id given by the “componentId” parameter. If the “targetCode” parameter is not supplied, the Widget Manager shall present the choice of migration targets to the user, by any appropriate means. If the “targetCode” parameter is supplied, the Widget Manager shall migrate the component widget to the designated target. The value of the target code shall be taken from a reply to the “requestMigrationTargets” message. The Widget Manager shall reply using a “returnCode” parameter whose value is: 1 for failure of the request and 0 for success.

```
<messageOut name="requestMigrationTargets">
    <input name="codes" scriptParamType="stringarray"/>
    <input name="names" scriptParamType="stringarray"/>
    <input name="descriptions" scriptParamType="stringarray"/>
</messageOut>
```

This message is sent by a Widget to request that the Widget Manager investigates what are the possible migration targets, i.e. devices a widget can be migrated to. The parameters “codes”, “names” and “descriptions” shall be arrays of string values, with the same number of values in each of the three. The “codes” parameter shall contain the list of codes to be used for a migration target in the message “migrateComponent” above. The “names” parameter shall contain the list of names of migration targets. The “descriptions” parameter shall contain the list of descriptions of migration targets.

Add 10.24:

10.24 The <mw:componentChoice> element

Description:

This element identifies a set of alternative components. Upon activation of the componentChoice, the widget manager shall try, one by one, to load the component children, in document order. As soon as one component child activation succeeds, the widget manager stops.

Context:

This is a child of a <content> element.

Expected children (in any order):

One or more <mw:component> elements, on which the attributes id, activateTrigger, deactivateTrigger, activatedAction, deactivatedAction, activateFailureAction and activationTarget are not allowed.